NAVAL POSTGRADUATE SCHOOL

MONTEREY, CALIFORNIA

THESIS

ENHANCEMENT OF THE ACQUISITION PROCESS FOR A COMBAT SYSTEM—A CASE STUDY TO MODEL THE WORKFLOW PROCESSES FOR AN AIR DEFENSE SYSTEM ACQUISITION

by

Wee Lee Chia

December 2009

Thesis Advisor:	James Bret Michael
Thesis Co-Advisor:	Man-Tak Shing

Approved for public release; distribution is unlimited

REPORT DOCUMENTATION PAGE			*Form Approved OMB No. 0704-0188*
Public reporting burden for this collection of information is estimated to average 1 hour per response, including the time for reviewing instruction, searching existing data sources, gathering and maintaining the data needed, and completing and reviewing the collection of information. Send comments regarding this burden estimate or any other aspect of this collection of information, including suggestions for reducing this burden, to Washington headquarters Services, Directorate for Information Operations and Reports, 1215 Jefferson Davis Highway, Suite 1204, Arlington, VA 22202-4302, and to the Office of Management and Budget, Paperwork Reduction Project (0704-0188) Washington DC 20503.			
1. AGENCY USE ONLY ***(Leave blank)***	**2. REPORT DATE** December 2009	**3. REPORT TYPE AND DATES COVERED** Master's Thesis	
4. TITLE AND SUBTITLE Enhancement of the Acquisition Process for a Combat System—A Case Study to Model the Workflow Processes for an Air Defense System Acquisition		**5. FUNDING NUMBERS**	
6. AUTHOR(S) Wee Lee Chia			
7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES) Naval Postgraduate School Monterey, CA 93943-5000		**8. PERFORMING ORGANIZATION REPORT NUMBER**	
9. SPONSORING /MONITORING AGENCY NAME(S) AND ADDRESS(ES) N/A		**10. SPONSORING/MONITORING AGENCY REPORT NUMBER**	
11. SUPPLEMENTARY NOTES The views expressed in this thesis are those of the author and do not reflect the official policy or position of the Department of Defense or the U.S. Government.			
12a. DISTRIBUTION / AVAILABILITY STATEMENT Approved for public release; distribution is unlimited		**12b. DISTRIBUTION CODE**	

13. ABSTRACT (maximum 200 words)

This thesis examines the challenges involved in the concept-refinement phase of the acquisition process of a complex system. We investigated the technical feasibility of using the Goal Question Metric methodology as part of the concept-refinement phase of the requirements analysis. Use case analysis and activity diagram modeling were used to analyze the workflow of a generic concept-refinement process. Statechart assertions and runtime execution monitoring were then used to formally specify the process and check for compliance to the process during its enactment.

14. SUBJECT TERMS Acquisition Process, Concept-Refinement Phase, GQM Method, Use Case Analysis, Activity Diagram, Workflow Process, StateChart Assertions, Runtime Execution Monitoring			**15. NUMBER OF PAGES** 67
			16. PRICE CODE
17. SECURITY CLASSIFICATION OF REPORT Unclassified	**18. SECURITY CLASSIFICATION OF THIS PAGE** Unclassified	**19. SECURITY CLASSIFICATION OF ABSTRACT** Unclassified	**20. LIMITATION OF ABSTRACT** UU

NSN 7540-01-280-5500 Standard Form 298 (Rev. 2-89)
Prescribed by ANSI Std. 239-18

THIS PAGE INTENTIONALLY LEFT BLANK

Approved for public release; distribution is unlimited

ENHANCEMENT OF THE ACQUISITION PROCESS FOR A COMBAT SYSTEM—A CASE STUDY TO MODEL THE WORKFLOW PROCESSES FOR AN AIR DEFENSE SYSTEM ACQUISITION

Wee Lee Chia
Lieutenant Colonel, Republic of Singapore Air Force (RSAF)
B.Sc. (Hons) Comp Sci, The Queen's University of Belfast, U.K., 1997

Submitted in partial fulfillment of the
requirements for the degree of

MASTER OF SCIENCE IN COMPUTER SCIENCE

from the

NAVAL POSTGRADUATE SCHOOL
December 2009

Author: Wee Lee Chia

Approved by: Professor James Bret Michael
Thesis Advisor

Professor Man-Tak Shing
Co-Advisor

Professor Peter Denning
Chairman, Department of Computer Science

THIS PAGE INTENTIONALLY LEFT BLANK

ABSTRACT

This thesis examines the challenges involved in the concept-refinement phase of the acquisition process of a complex system. We investigated the technical feasibility of using the Goal Question Metric methodology as part of the concept-refinement phase of the requirements analysis. Use case analysis and activity diagram modeling were used to analyze the workflow of a generic concept-refinement process. Statechart assertions and runtime execution monitoring were then used to formally specify the process and check for compliance to the process during its enactment.

THIS PAGE INTENTIONALLY LEFT BLANK

TABLE OF CONTENTS

THIS PAGE INTENTIONALLY LEFT BLANK

LIST OF FIGURES

THIS PAGE INTENTIONALLY LEFT BLANK

LIST OF TABLES

THIS PAGE INTENTIONALLY LEFT BLANK

LIST OF ACRONYMS AND ABBREVIATIONS

AoA	Analysis of Alternatives
ICD	Initial Capability Document
MDA	Milestone Decision Authority
PM	Project Manager
TDS	Technology Development Strategy
CDD	Capability Development Document
OT&E	Operational Testing and Evaluation
IOC	Initial Operational Capability
BPM	Business Process Modeling
BPMN	Business Process Modeling Notation
SoA	Service-oriented Architecture
UML	Unified Modeling Language
CSP	Communicating Sequential Protocol
UCDMO	Unified Cross Domain Management Office
IDE	Integrated Development Environment
GQM	Goal, Question and Metric
DR	Derived Requirement
CLAWS	Complementary Low-Altitude Weapon System
USMC	United States Marine Corps
CM	Cruise Missile
UAV	Unmanned Aerial Vehicle
FW/RW	Fixed Wing/Rotary Wing
HMMWV	High-Mobility Multipurpose Wheeled Vehicle
AMRAAM	Advanced Medium-Range Air-to-Air Missile
LAAD	Low-Altitude Air Defense
GOTS	Government Off-The-Shelf
COTS	Commercial Off-The-Shelf
GFE	Government Furnished Equipment
NDI	Non-Developmental Item
C3I	Command, Control, Communications and Intelligence

O&S	Operations and Support
AD	Activity Diagram
V&V	Verification and Validation

ACKNOWLEDGMENTS

I would like to extend my greatest gratitude to the Republic of Singapore Air Force for investing and providing me this opportunity to immerse myself academically in such a great institution.

I want to thank my thesis advisor, Professor Bret Michael, for introducing me to this body of knowledge. From this positive learning experience, I believe the knowledge gained is something that I can anchor on and pursue beyond the scope of this thesis.

I also want to thank Professor Man-Tak Shing for his relentless effort in guiding me in my research and the successful completion of this thesis. I am especially grateful to his patience and guidance despite the many challenges encountered in researching for this thesis.

Last, but not least, I want to thank my wife and my two lovely children for their love, understanding and emotional support, without, which I am pretty sure it would be an uphill task to juggle my time between family and studies.

THIS PAGE INTENTIONALLY LEFT BLANK

I. INTRODUCTION

A. BACKGROUND

In the acquisition process of a complex system, the goal is to have a process that produces defensible decisions supported by sound analyses and clear rationale. With ever stretched defense budgets and the push in the U.S. for reform in defense acquisition in the form of minimizing the number of requirements to be specified by military specifications, the design trade space has to be broadened enough for examination of a full range of alternatives by the developing contractor. As a result, the pre-acquisition phase, in which system concepts are refined and technology maturity and capabilities are identified, plays a vital role in the successful acquisition of today's complex systems. This phase, also known as the concept-refinement phase in the U.S. DoD acquisition process,[1] preludes the actual initiation of the program.

Project Managers (PM) need to consider system complexity, the aggregation of emerging and mature technologies, system interoperability, and system evolution. Besides these operational considerations, the project team must also consider organizational issues. Some of these issues will differ from one acquisition program to another, but there are issues that these programs share, such as those associated with the assessment of risk and cost-benefit analyses, as well as acceptance and evaluation. Other considerations include the type of acquisition approach to adopt (for example, incremental, evolutionary or agile), external interfaces, use of previously developed or commercial software, competition/solicitation approach, contracting approach, information assurance strategy, training, and support.[2] The challenges encountered during pre-acquisition can go beyond requirements engineering, and operations to include development strategy and sustainment. A lot has to do with human-in-the-loop processes

[1] Department of Defense (DoD), Instruction 5000.2, "Operation of the Defense Acquisition System," May 12, 2003.

[2] Carnegie-Mellon University, "Acquisition Overview: The Challenges," https://buildsecurityin.us-cert.gov/daisy/bsi/articles/best-practices/acquisition/893-BSI.html (accessed December 8, 2009).

and at times, to a certain extent, venturing into 'unchartered territory'. In contrast, the other phases of the acquisition process are typically better developed as more research is conducted and experience gained for such processes. As an example, in a system development phase, developers or PM can depend on an established model such as the Capability Maturity Model Integration (CMMI).[3] The model covers the acquisition discipline as it relates to product development and maintenance, which includes the use of commercial off-the-shelf (COTS) products and outsourcing. As a result, developers and PM tend to be pre-disposed to use well-established models, with the perception that the challenges faced would not be so varied and critical than when using non-standard model. However, the same cannot be said for pre-system acquisition as not much formal work has been conducted in this area. To address this gap and to meet the challenges, process modeling and methodology for formally capturing acquisition processes must be in place.

The objective of this thesis was to investigate ways to improve the concept-refinement tasks in the pre-system acquisition process. The scope of the research is to propose a methodology that can be used to derive measurable requirements for an acquisition program to define the success of the system acquisition, and to apply existing techniques to assure that the methodology is carried out as specified.

In the acquisition of software-intensive systems, how can stakeholders specify their program requirements in a non-ambiguous way to system designers? More often than not, the vagueness and ambiguity of natural-language specifications of requirements can be easily misinterpreted by the engineers developing a system. Consequently, the actual requirements performed may differ from that of the intent of the stakeholders. On the other hand, if engineers represent requirements in a formal (that is, mathematical) language with a restricted vocabulary, stakeholders may not fully appreciate the logic of the specifications due to lack of training or knowledge of formal methods. Hence, a 'bridging of minds' between stakeholders and developers is desired especially in the development of complex systems.

[3] Software Engineering Institute, Carnegie-Mellon University, "Capability Maturity Model Integration (CMMI)," http://www.sei.cmu.edu/cmmi/start/faq/related-faq.cfm (accessed December 8, 2009).

As a step toward accomplishing the bridging, the author investigated the well-established Goal, Question, Metrics (GQM) method by applying it to the concept-refinement phase of the acquisition of a combat system, known as the Complementary Low-Altitude Weapon System (CLAW) for low-altitude air defense capability.

The analysis of CLAW involved creating use cases and modeling the workflow of the concept-refinement phase with activity diagrams. The author then uses these artifacts to guide the formalization of the workflow requirements and the author settled on using Statechart assertions. The author then demonstrated the technical feasibility of exploying assertion-based validation and executable runtime monitoring of the workflow, with the aim of ensuring the correct enactment of the process.

B. ORGANIZATION

The organization of this thesis includes an introduction, three development chapters, and a final chapter for conclusions and future work. Chapter II presents an overview of the acquisition process, along with a discussion of the need for precise workflow process modeling. In Chapter III, the author introduces the GQM method as a means to develop measurable program requirements and provides details of this process through the employment of a methodology using a case study for the acquisition system. Chapter IV describes the modeling of an acquisition process in terms of Unified Modeling Language (UML) activity diagrams and the use of Statechart assertions to check for the proper and timely execution of the acquisition tasks in the acquisition process model. Chapter V contains conclusions and topics for future work.

THIS PAGE INTENTIONALLY LEFT BLANK

II. THE ACQUISITION PROCESS

A. OVERVIEW

One of the key purposes of DoDI 5000.2 is to "establish a simplified and flexible management framework for translating mission needs and technology opportunities, based on approved mission needs and requirements, into stable, affordable, and well-managed acquisition programs that include weapon systems and automated information systems (AISs)."[4] It defines three key processes that must work in concert to deliver the capabilities required by the warfighters: the requirement process (Joint Capabilities Integration & Development System [JCIDS]), the acquisition process (Defense Acquisition System), and program and budget development (Planning, Programming, Budgeting, and Execution [PPBE] process).[5]

This chapter contains an overview of the Defense Acquisition Management Framework, followed by a review of process-modeling approaches identified via a search of the open literature. The intent is not to dive into the details of the steps in the process but to provide a macro view of the mechanisms involved.

B. THE DEFENSE ACQUISITION MANAGEMENT FRAMEWORK

The U.S. defense acquisition process is structured into discrete phases separated by major decision points (called milestones or decision reviews) with a number of key activities to provide the basis for comprehensive management and informed decision-making.[6] This is based on the policy of:[7]

1. Flexibility - tailoring program strategies and oversight to fit the particular conditions of the program

---

[4] DoD, Instruction 5000.2.

[5] Defense Acquisition University (DAU), "Integrated Defense Acquisition, Technology, and Logistics Life Cycle Management System," June 15, 2009, https://acc.dau.mil/IFC/index.htm (accessed November 11, 2009).

[6] Ibid.

[7] DoD, Directive 5000.1, "The Defense Acquisition System," May 12, 2003.

2. Responsiveness - rapid integration of advanced technologies through evolutionary acquisition

3. Innovation - adoption of practices that reduce cycle time and cost, as well as encourage teamwork

4. Discipline - use of approved program baseline parameters as control objectives, identifying deviations, and exit criteria

5. Streamlined and effective management – project members are empowered with sufficient authority while maintaining accountability

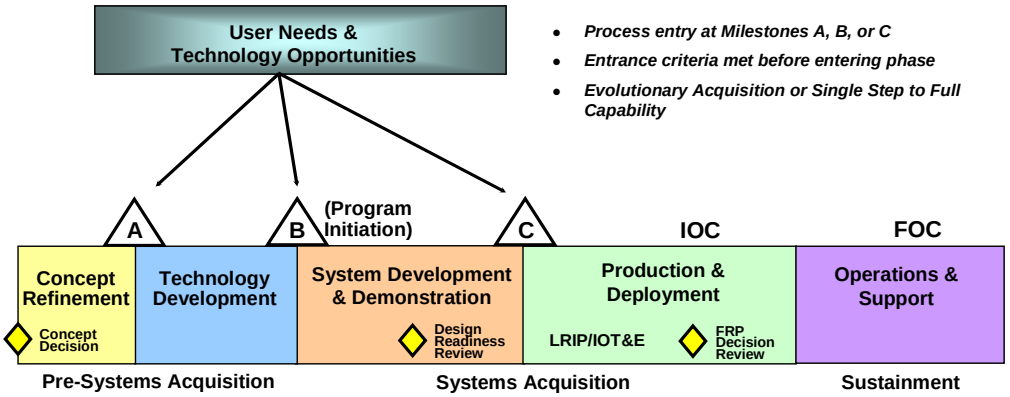

Figure 1. The Defense Acquisition Management Framework[8]

The acquisition process begins with the identification of a capability need that requires a material solution. The process encompasses the activities of design, fabrication, test, manufacturing, operations, and support. It may involve modifications, and it ends with disposal/recycling/demilitarization. Major upgrade or modification programs may also follow the acquisition life cycle process.[9]

There are five major phases in this process,[10] namely concept refinement, technology development, system development and demonstration, production and deployment, as well as operations and support. Within this process, the first two are part of the pre-acquisition phase. This is because it is only after phase two that the new acquisition program is initiated.

[8] Department of Defense (DoD), Instruction 5000.2, "Operation of the Defense Acquisition System," May 12, 2003.

[9] DAU, "Integrated Defense Acquisition."

[10] DoD, Instruction 5000.2.

The first phase is concept refinement. The purpose is to refine the initial concept that was provided as part of the input into this phase. The other inputs that are typically accompanied by this process are the Analysis of Alternatives (AoA) plan, exit criteria as well as the alternative maintenance and sustainment concepts. It is a very rigorous process as the team has to analyze the operational capabilities and environmental constraints, perform trade-off studies to assess critical technologies associated with the operational concepts, and develop exit criteria for the various phases, as well as the test plans. The phase ends when the Technology Development Strategy (TDS) document is developed and the Milestone Decision Authority (MDA) approves the preferred solution presented in the TDS document. In Chapter III, a GQM template is applied as an approach to develop some of these decisions.

Technology development is the second phase, and the purpose is to reduce the technological risk of the program by determining the appropriate set of technologies to be integrated into a full system. It is an iterative process to assess the viability of the various technologies as spelled out in the TDS document. As such, it is inevitable that there is close collaboration among the developers, the science, engineering and technology community, as well as the user during this phase. As there are various technologies to be considered for different applications in a system, it is possible to obtain incremental approval by the MDA once a proposed solution has demonstrated itself to be militarily useful. The phase ends when all the increments of the militarily useful capability have been identified and successfully demonstrated. The phase can also end if the MDA terminates the program. If approval is given by the MDA to proceed with program initiation, the Capability Development Document (CDD), which spells out the details on operational performance to design the proposed system, will be provided as an input for the next phase.

The system development and demonstration phase marks the initiation of the acquisition program. While the system is being developed in accordance with the CDD, the emphasis is to ensure operational supportability and minimize the logistics footprint. Other factors that are considered in this phase include ensuring affordability and protection of critical program information and demonstrating system integration,

interoperability, and safety. To obtain approval for the MDA to commit into the program, one of the focal points is system demonstration. This effort is conducted "when a system is demonstrated in its intended environment, using the selected prototype; meets approved requirements; industrial capabilities are reasonably available; and the system meets or exceeds exit criteria."[11]

The purpose of the production and deployment phase is to "achieve an operational capability that satisfies mission needs."[12] To ensure this, the system is typically subjected to numerous operational test and evaluation (OT&E) activities to determine the effectiveness and suitability of the system prior to the commencement of this phase. As a result of this effort, low-rate initial production will commence

> to ensure adequate and efficient manufacturing capability and to produce the minimum quantity necessary to provide production or production-representative articles for IOT&E, establish an initial production base for the system; and permit an orderly increase in the production rate for the system, sufficient to lead to full-rate production upon successful completion of operational (and live-fire, where applicable) testing.[13]

Additionally, life-cycle support issues, such as training, maintenance and support, and upgrades should be re-examined and updated accordingly. One key outcome arising from this phase is the declaration on the attainment of Initial Operational Capability (IOC) by the system user.

The fifth phase, also known as the sustainment phase, addresses operations and support. The purpose is to establish support to the system, which entails issues such as maintenance, upgrades, training, and technical support. This is performed throughout the life cycle of the system so that it can be managed in the most cost-efficient manner while maintaining the system's operational needs. Key decisions that are made during this phase address the affordability of the mixing of systems, as predictions about costs become reality. In addition, as threats change it is possible that users will move to retire certain systems and procure additional numbers of other existing systems.

---

[11] DoD, Instruction 5000.2.

[12] Ibid.

[13] Ibid.

What is key in this framework is the provision of the necessary documents as inputs from one phase to another, for example, the AoA plan and Initial Capability Document (ICD), as inputs to the concept-refinement phase, before the phase can begin. Additionally, the approval required by a MDA before advancing to the next phase is a form of governance imposed on this framework. Hence, the adherence to this framework provides the PM with some level of confidence for the development of his or her system. It is also not a rigid framework as it allows the PM and the MDA to exercise "discretion and prudent business judgment"[14] to tailor the number of phases and decision points for the specific needs of their program.

Due to the complexity involved in the development of large-scale systems, it is common for the program to be divided into smaller projects that is managed by the PM. To ensure success, it is imperative that the PM and his team members design the framework to reflect the demands of the program and define the required inputs, the goals for each phase of the process, and the required outputs to ramp up the next phase.

C. PROCESS MODELING

Process modeling is a key concept in process engineering. The model forms the basis for the actual process for use in the development of a system. It is a model to describe the actions of processes of the same nature. However, the model is just an abstraction that requires more detail to be filled in to derive the actual process. It is not a precise representation of a process on what actually happened but rather a rough anticipation of how the process should behave. Hence, the model has to be refined for it to work. The goals of a process model are as follows:[15]

[14] DoD, Instruction 5000.2.

[15] Wikipedia, "Process Modeling," http://en.wikipedia.org/wiki/Process_modeling, (accessed December 8, 2009).

1. Descriptive
    a. Track what actually happens during a process.
    b. Takes the point of view of an external observer who looks at the way a process has been performed and determines the improvements that have to be made to make it perform more effectively or efficiently.
2. Prescriptive
    a. Defines the desired processes and how they should/could/might be performed.
    b. Lays down rules, guidelines, and behavior patterns, which, if followed, would lead to the desired process performance. They can range from strict enforcement to flexible guidance.
3. Explanatory
    a. Provides explanations about the rationale of processes.
    b. Explores and evaluates the several possible courses of action based on rational arguments.
    c. Establishes an explicit link between processes and the requirements that the model needs to fulfill.
    d. Pre-defines points at which data can be extracted for reporting purposes.

Much work has been conducted in the field of process modeling. There exists workflow technology to capture business processes as workflow specifications and increased workflow automation in complex real-world environments involving heterogeneous, autonomous, and distributed information systems.[16] In conjunction with this, Business Process Modeling (BPM) is used to represent processes in an enterprise. It is used to analyze current processes in order to improve business efficiency and quality. The advent of visual modeling language to represent business processes has proven effective for presenting it to business stakeholders, including business analysts and

[16] G. Diimitrios, H. Mark, and S. Amit, "An overview of workflow management: From process modeling to workflow automation infrastructure," *Distributed and Parallel Databases* 3, no. 2 (April 1995): 119-153.

system developers. Supporting technologies include Business Process Modeling Notation (BPMN), Unified Modeling Language (UML), model-driven architecture, and service-oriented architecture (SoA).[17]

There are also formal languages to provide a means of specifying the security properties of complex systems and protocols. Communicating Sequential Protocol (CSP) is such a language for describing process interaction in concurrent systems.[18] It is generally applied in industry as a tool for specifying and verifying the concurrent aspects of a variety of different systems and has seen active work to increase its range of practical applicability, for example, increasing the scale of the systems that can be tractably analyzed.[19] For the application of security properties software and the analysis of those properties in the context of a well-defined programming environment, there are formal approaches developed to find bugs and verify the absence of bugs in software.[20]

In the area of software tools, there are numerous developments to automate solutions to support process design and operation. The Alloy Analyzer, developed by researchers at the Massachusetts Institute of Technology (MIT), is one such tool that can be used to analyze specifications written in the Alloy specification language.[21] The Analyzer can generate instances of model invariants, simulate the execution of operations defined as part of the model, and check user-specified properties of a model. As the tool supports incremental analysis of the models, it is capable of generating immediate feedback to users as they are being constructed–a term known as analysis of partial models.[22] As a result, it can perform incremental analysis of models as they are constructed and provide immediate feedback to users. In FUNSOFT nets, an approach for

---

[17] Wikipedia, "Process Modeling."

[18] A. W. Roscoe, *The Theory and Practice of Concurrency* (Great Britain: Prentice Hall Europe, 1997).

[19] S. Creese, "Data Independent Induction: CSP Model Checking of Arbitrary Sized Networks," (D.Phil. thesis, Oxford University, 2001).

[20] H. Chen and D. Wagner, "MOPS: An infrastructure for examining security properties of software," *CCS '02: Proceedings of the 9th ACM Conference on Computer and Communications Security*, (2002): 235–244.

[21] D. Jackson, *Software Abstractions: Logic, Language, and Analysis* (MIT Press, 2006).

[22] Wikipedia, "Alloy Analyzer," http://en.wikipedia.org/wiki/Alloy_Analyzer, (accessed December 8, 2009).

modeling and analyzing software processes was introduced. It was based on Petri nets that were developed to support software process modeling. This was supposed to address the shortcomings of Petri nets that failed to "measure up to more detailed requirements such as tight but still comprehensible representations of software process models."[23]

In another modeling approach, the combination of UML and embedded Statecharts was applied to the Unified Cross Domain Management Office's (UCDMO) cross-domain solutions workflow process.[24] The technique of using Statecharts for the specification and development of a complex reactive system was used to formally specify and reason about the workflow in the UCDMO. In this approach, the StateRover[25] was used as a modeling tool for the workflow processes and the concept of embedded assertions applied to the workflow to check that specific requirements of the process were adhered to. With this development, the process engineer can formally reason the processes and conduct inspection and analysis of processes in a largely human-based workflow of the UCDMO.

For this thesis, the author models the workflow of the concept-refinement phase using an UML activity diagram. The author also utilizes UML-like Statechart assertions to specify the proper and timely execution of the tasks layout in the workflow model. The Statechart assertions are developed using StateRover for the Eclipse integrated development environment (IDE), which supports the creation and validation of the Statechart assertions via simulated test scenarios within the JUnit test framework.

---

[23] W. Emmerich and V. Gruhn, "FUNSOFT nets: a Petri-net based software process modeling language," *Software Specification and Design, Proceedings of the Sixth International Workshop* (1991): 175–184.

[24] M. A. Schumann, "A Statechart Model of the Cross Domain Implementation Process," *IATAC Newsletter* 12 (February 2009): 26–30.

[25] D. Drusinsky, *Modeling and Verification Using UML Statecharts A Working Guide to Reactive System Design, Runtime Monitoring and Execution-based Model Checking* (Burlington, MA: Elsevier, 2006).

III. THE GQM METHOD FOR CONCEPT REFINEMENT

A. INTRODUCTION

The intent of this chapter is to introduce the GQM approach as a method to develop measurable requirements of the project in the concept-refinement phase. The author shall illustrate the GQM method with a case study involving the acquisition of a low-cost, high-mobility, advanced low-altitude missile capability. Please note that while the case study involves an actual air defense system, the specifications, performances, and so on are not specific to the system per se. It is used solely to provide a more meaningful illustration of the application of the GQM model.

B. DESCRIPTION OF THE METHOD

GQM is a top-down, goal-driven approach, which ensures that all metrics are selected for a goal-driven purpose. It is particularly useful, for example, in requirements definition, because every requirement should be measurable. In the event that it cannot be measured, either the user has to review and redefine it or that particular requirement will not be included in the specification document. This is pertinent as it safeguards both the interest of the user and developer when the system is finally fielded and all requirements are measured to meet the specification. In essence, what cannot be measured should not be considered a requirement.

The GQM method was developed as a measurement mechanism for feedback and evaluation. It supports program planning, for example, determining the cost of a new program, evaluating the strengths and weaknesses of the current processes and products, adopting/refining techniques as well as evaluating the quality of specific processes and products. Measurement also helps, during the course of a program, to assess its progress, take corrective action based on this assessment, and evaluate the impact of such action.[26]

[26] V. R. Basili, "Data Collection, Validation, and Analysis," *Tutorial on Models and Metrics for Software Management and Engineering*, IEEE Catalog no. EHO-167-7 (1981): 310–313.

In order to improve a process, the organization must align their measurement goals to corporate goals. These goals can then be translated to activities that can be definitively measured.

As shown in Figure 2, there are four phases to the GQM method:[27]

1. The planning phase is performed to fulfill all basic requirements to make the GQM measurement program a success. This includes training, management involvement and project planning, resulting in a project plan.
2. In the definition phase, all deliverables are developed (goals, questions, metrics and hypotheses are defined) with emphasis on structured interviews and knowledge acquisition techniques. Actual measurements can commence when the definition activities are completed.
3. In the data collection phase, all the collected data are properly stored in a measurement database.
4. In the interpretation phase, the measurement database is utilized to answer the stated questions. These answers are used again to determine if the stated goals can be achieved.

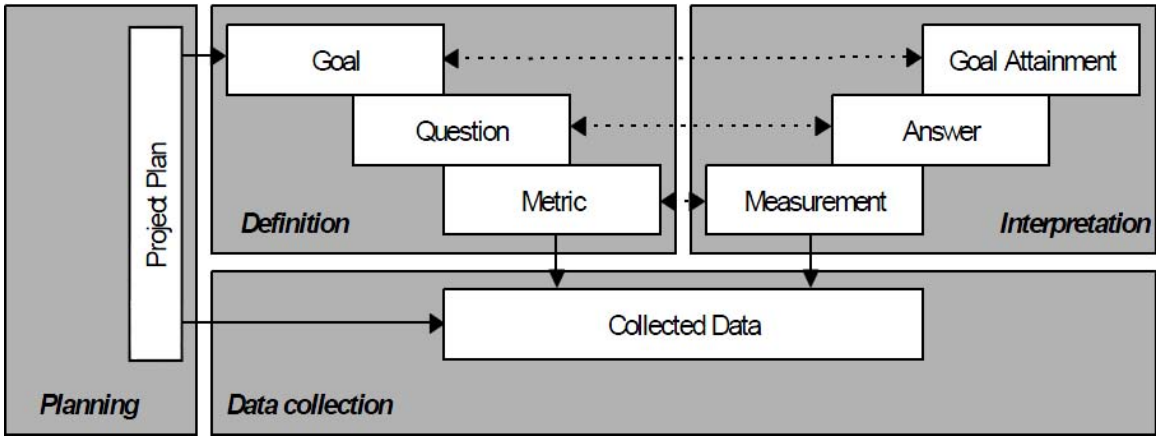

Figure 2. The Four Phases of GQM

Figure 3 shows the hierarchical structure of a GQM model, which has the following three levels:[28]

1. Conceptual level (goal) - A goal is defined for an object for a variety of reasons, with respect to various models of quality, from various points of view and relative to a particular environment.

[27] Rini Solingen and Egon Berghout, *The Goal/Question/Metric Method: A Practical Guide for Quality Improvement of Software Development* (McGraw-Hill, 1999), 22–23.

[28] Ibid.

2. Operational level (question) - A set of questions is used to define models of the object of study and then focuses on that object to characterize the assessment or achievement of a specific goal.

3. Quantitative level (metric) - A set of metrics, based on the models, is associated with every question in order to answer it in a measurable way.

It is intuitive in the sense that all questions should begin with a goal in mind. Several questions can be raised whereby metrics are developed for each question to measure the outcome. Several goals can also have questions and metrics in common, which ensures that when the measure is actually taken, the different viewpoints are taken into account correctly; that is, the metric might have different interpretations when taken from different viewpoints.[29] GQM structures of goals, questions and metrics should be built upon the knowledge of the experts in the organization. Particularly in the definition phase, where knowledge acquisition techniques are also applied to capture the implicit models of the developers built during the years of experience. Those implicit models give valuable input into the measurement program and will often be more important than the available explicit process models.[30]

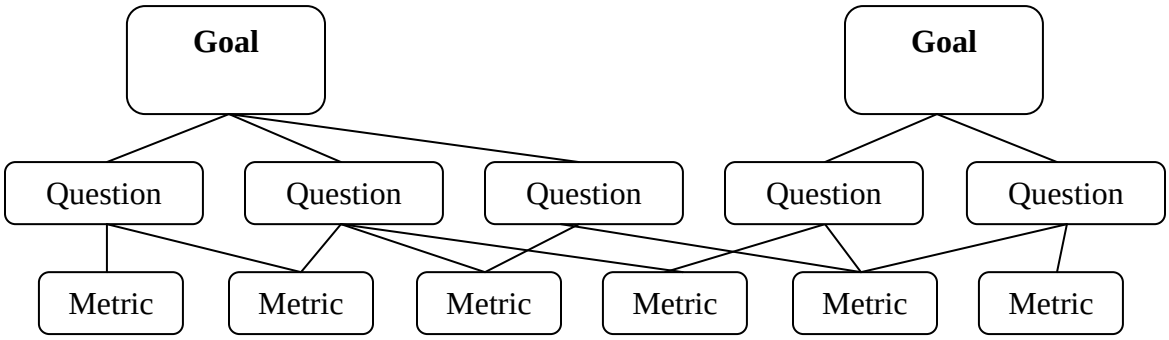

Figure 3. Hierarchical Structure of a GQM Model

[29] Basili, "Data Collection."

[30] Solingen and Berghout, *The Goal/Question/Metric Method*, 22–23.

Another area to highlight is the evolution of GQM to include models of software processes and products. The result is a model-based GQM approach[31] that defines metrics into two perspectives, namely, metrics definition by members of the project team using GQM method and metrics definition based on models of software processes and products, as shown in Figure 4. The intent is to crosscheck both metrics for consistency and completeness. Once this is completed, the actual GQM plan is developed. The GQM plan consists of the steps in a measurement program to document the goals, related questions, and identified metrics. Measurement can start once the plan is approved. Data are collected on the development process and products, aggregated, and validated. Finally, the measurement results are returned to members for analysis, interpretation, and evaluation based on the GQM plan.[32]

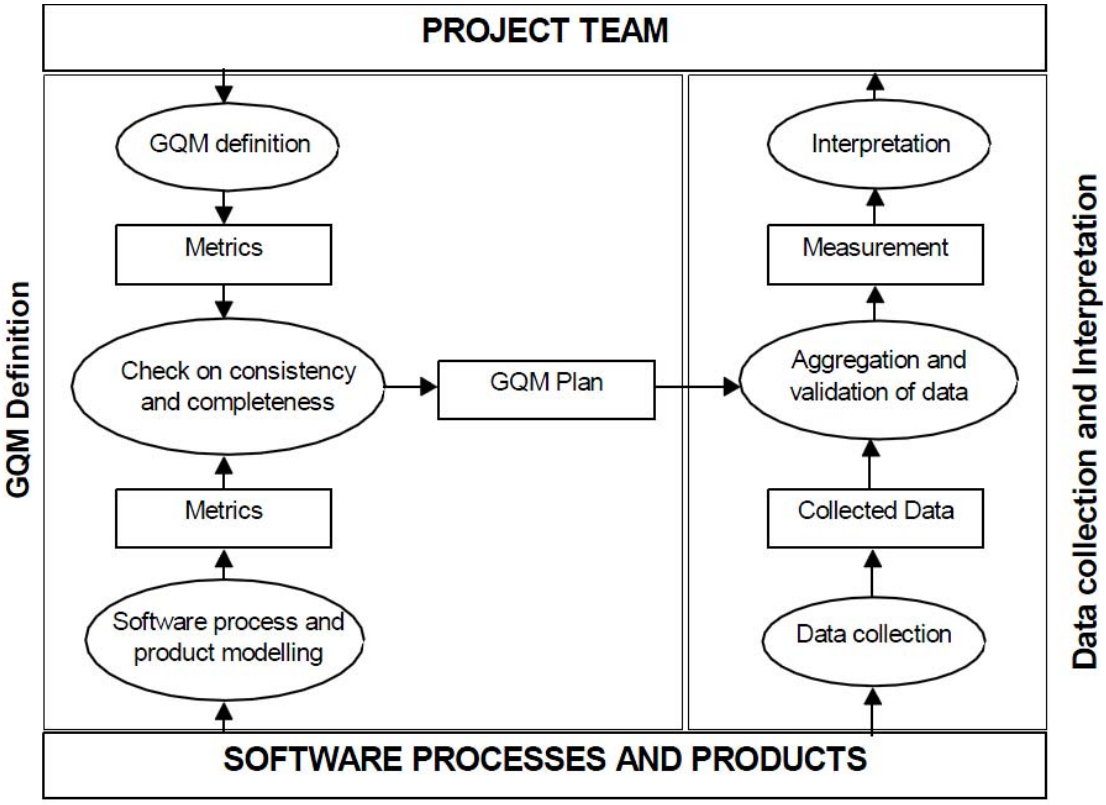

Figure 4. Metrics Modeling from two Perspectives

[31] Solingen et al, "Application of Software Measurement at Schlumberger RPS: Towards Enhancing GQM," *Proceedings of the 6th European Software Control and Metrics (ESCOM) Conference*, May 17–19, 1995.

[32] Ibid.

C. THE CLAW AIR DEFENSE SYSTEM[33]

The Complementary Low-Altitude Weapon System (CLAWS) is operated by the United States Marine Corps (USMC) to achieve combat effectiveness over large sectors of the battlespace. The system, which is to be highly-mobile and can be deployed rapidly, is expected to possess high firepower, be all-weather capable, and be equipped with standoff capabilities to effectively engage current and emerging air threats. The types of threats include cruise missiles (CM), unmanned aerial vehicles (UAV), and advanced fixed-wing/rotary-wing (FW/RW) aircraft.

The system consists of the M1097 High-Mobility Multipurpose Wheeled Vehicle (HMMWV) with a minimum of four Advanced Medium-Range Air-to-Air Missile (AMRAAM) missiles mounted on a launcher. The CLAW, with its extended range, lethality and accuracy, is meant to be operated by a crew of two. The concept of operation is to complement the current Low-Altitude Air Defense (LAAD) battalion that operates the Stinger system. This added capability will provide enhanced protection, beyond the range and capability of existing systems in the LAAD battalion. The new capability will also maintain the high-mobility required for organic protection of maneuver elements in the USMC.

One of the main developmental considerations for the CLAW program is to maximize the use of current Department of Defense (DoD) military equipment. This will ensure the program can be managed in a cost-effective manner and meet short delivery timelines. As such, it is expected that the program will take full advantage of government off-the-shelf (GOTS) products, commercial off-the-shelf (COTS) products, government furnished equipment (GFE), and other types of non-developmental items (NDI) to meet this stringent requirement. Evident from this requirement is the adaptation of the AMRAAM system, which was originally developed as an air-to-air missile. Likewise, the platform in consideration to house the entire system is the HMMWV vehicle, which is used by the Avenger Air Defense System, also operated by the LAAD battalion.

---

[33] GlobalSecurity.org, http://www.globalsecurity.org/military/systems/munitions/claws.htm (accessed November 15, 2009).

Figure 5. The CLAWS (HUMRAAM) Surface-launched AMRAAM Air Defense Missile[34]

Besides short development time and low-risk concerns (especially budgetary concerns), one of the key advantages for such development consideration is that the systems are all fielded systems and combat-proven. While it is based on mature technology, the capability is sufficient to meet the specified threats. Additionally, the developed system can easily integrate with current Command, Control, Communication and Intelligence (C3I) architecture and interface with the suite of available sensors for full air-situation awareness. In short, it is a system that is ready to be deployed for action once it is delivered without concerns of the many initial problems that often beset systems based on new technologies. The Operations and Support (O&S) phase of the program is also expected to be more efficient and supportable due to the economies of scale in sharing similar components and expertise.

D. APPLICATION OF THE GQM METHOD TO THE ACQUISITION SYSTEM

This section presents an application of the GQM method to the development of the CLAW system. The method is useful in that it helps to provide better requirements analysis. Referring back to the acquisition cycle in Chapter II, the output of this analysis will serve as part of the formulation of the TDS document, which then serves as the input

[34] Net Resources International, "Surface-Launched AMRAAM (SL-AMRAAM / CLAWS) Medium-Range Air Defense System, USA," http://www.army-technology.com/projects/surface-launched/surface-launched1.html (accessed November 15, 2009).

to the technology development phase. Specifically, the stated requirement which is extracted from the ICD is to develop a low-cost, high-mobility, advanced low-altitude missile capability. The intent is to develop strategies and measurable metrics that will answer directly to the stated requirement. Measurements also help during the course of a project to assess its progress, take corrective action based on this assessment, and evaluate the impact of such action. As stated in an earlier chapter, the figures used are fictitious and are used solely to provide a more meaningful illustration of the application of the GQM methodology.

The GQM method, as recommended by Basili and Rombach,[35] begins with a framework for the construction of goals. In the design of an actual acquisition system, there would be many strategies for such a requirement. However, as an illustration to demonstrate this approach, only three strategies are established as shown in Figure 6. As described and shown in the figure, the CLAW requirement was to develop a low-cost, high-mobility, advanced low-altitude missile capability. In order to satisfy this requirement, three strategies were developed as follows:

> S1 : To maximize the use of existing technologies and DoD military equipment. This will ensure the program can be managed in a cost-effective manner and meet short delivery timelines.
>
> S2 : To meet the mobility requirement for the system to keep pace with expeditionary and supported maneuver elements.
>
> S3 : To maximize the use of NDI including an unmodified AMRAAM missile.

With these three strategies, goals were then developed for each of the strategies. The number of goals varied depending on the complexity of the strategies. In this example, four goals were set to meet the strategies for the requirement. For the first strategy (S1), the goal (G1) was to adapt from the existing LLAD support system. However, it is not possible that all of the existing LLAD support system is compatible to support the newly developed system. Hence, the question (Q1) was raised with regard to what specific types of support/maintenance systems were available to meet this goal. As a

[35] V. R. Basili and H. D. Rombach, "The TAME Project: Towards Improvement-Oriented Software Environments," *IEEE Transactions on Software Engineering* SE-14, no. 6 (June 1988): 761.

means to ensure that the strategy was sufficiently accomplished, the metric (M1) for this goal was the extent (in terms of percentile) that current DoD inventory was used.

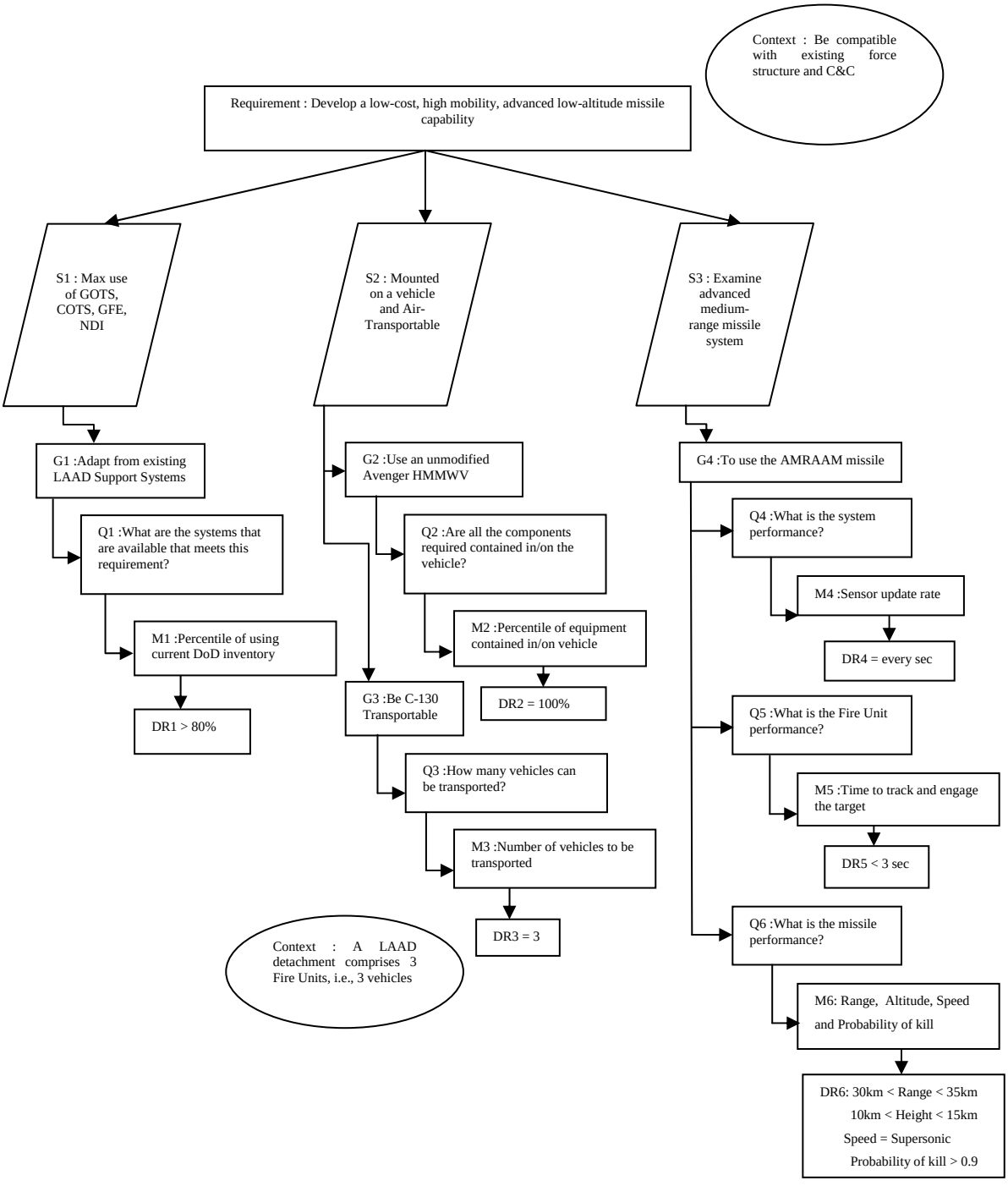

Figure 6. The CLAWS GQM Model

For the second strategy (S2), the goal (G2) was to use an unmodified Avenger HMMWV. For the vehicle to be suited for the newly developed system, the question (Q2) that was raised dealt with ensuring that the HMMWV is completely self-sustained such that it is capable of carrying all equipment/personnel needed for its mission. The metric (M2) for this goal was the extent (also in terms of percentile) that the equipment can be contained in the vehicle. Another goal (G3) was set to further support this strategy. The goal was to meet mobility needs, and in this case, be air-transportable to meet mission requirements. The question (Q3) that was raised was concerning the accommodation of the vehicles into a C-130 transport plane, with the metric (M3) for this goal being the number of vehicles that was required to fit into the C-130 transport plane.

For the last strategy (S3), a decision was made earlier to adopt a NDI approach, and hence the AMRAAM missile was chosen as the goal (G4) to fulfill this strategy. However, as the AMRAAM missile was originally developed for air-to-air combat missions, there was a need to ascertain that it can be successfully modified for land-based, air defense mission. The three questions (Q4, Q5 and Q6) that were asked were made to ensure that the AMRAAM missile for the CLAW system can be modified successfully to complement the LLAD force structure and integrated seamlessly to the existing C3I infrastructure. Consequently, three metrics (M4, M5 and M6) were used as unit of measurement for the required performance parameters for this goal.

The metrics only provide the unit of measurement for each stated goal. To ensure the requirements were quantifiable, they were further refined into derived requirements (DR) based on the metrics. The respective DRs for the metrics are shown in Figure 6.

As can be seen in the figure, contextual information was provided where necessary so that project team members could relate to the intent of the requirement for them to exercise better analysis of the program. In this case, the context for this requirement was for the developed system to be compatible with the existing force structure and C3I infrastructure. With this context, the project team would then be able to scope their development strategies, taking into consideration interoperability and integration issues.

For instance, to illustrate the role played by contextual information, it was written in earlier paragraph that there was one goal (G1) to meet strategy S1, and it was to adapt from existing LAAD support systems. Relating from the earlier context of being compatible with the existing force structure, this goal was chosen not only to meet strategy S1, which was to maximize the use of existing technologies and equipment so as to minimize cost and the delivery schedule, but also to ensure compatibility in terms of maintenance, training, spares support, etc. to the existing systems in the LAAD combat and service support structure. Achieving this would further ensure reduced O&S costs to the life cycle of the system. This exemplified the need for the project team to know the context of the program beyond the capabilities and performance parameters of the combat system. In another example, the goal (G3) for strategy S2 was to be C-130 transportable. The question raised was to ascertain the number of vehicles to be transported. In this case, the derived requirement (DR3) indicated three vehicles were sufficient. It might be trivial as this metric could be easily obtained by checking with the operators of the system. However, the context given to the project team was that a LAAD fighting unit comprised three fire-units. Hence, the C-130 must be able to accommodate this number. However, beyond that with this context the project team could also bear in mind this configuration in other analysis. In essence, their development should be done with the correct context in mind and not in isolation.

In summary, the GQM method supports the process to work systematically downwards where questions can be raised to achieve more clarity. Together, the questions and metrics are identified to fulfill the goal. The objective is to produce measurable metrics. These metrics can then be further refined with the stakeholders (for example, system operators) and established as a form of a measure of performance during the OT&E as part system and development phase of the program.

IV. ACQUISITION PROCESS MODELING AND ASSURANCE

A. INTRODUCTION

In this chapter, the UML artifacts and the Statechart assertions were used as light-weight formal approaches to model and verify the correct behavior of the system. To achieve this, the UML activity diagram was used as a semi-formal approach to model the workflow process of the concept-refinement phase in an acquisition process. It is a suitable analysis tool for this purpose as it can be used to describe the workflow in varying levels of detail.[36] These details can then be used to construct Statechart assertions using the StateRover tool[37] to specify formal assertions and to ensure the proper and timely execution of the acquisition tasks layouts in the acquisition process model.

B. ACQUISITION PROCESS MODELING

1. Use Case Analysis

A use case in software engineering and systems engineering is a description of a system's behavior as it responds to an event that originates from outside of that system. In other words, it is a scenario to illustrate some sequence of interactions between a user and a system so as to capture the system's behavioral requirements by detailing scenario-driven processes through the functional requirements.[38] "The use cases help the modelers understand the problems to be solved and the objectives to be accomplished by the perceived system. The high-level use cases are goal-oriented, and typically are used to describe the workflow of a business process instead of interactions among system components. Mapping the scenarios of the use cases to activity diagrams helps highlight the assignment of responsibilities and the interdependencies among the different components (of an organization or system)".[39]

---

[36] Simon Bennett, John Skelton and Ken Lunn, *Schaum's Outlines of UML* (McGraw-Hill International, 2001), 208.

[37] Drusinsky, *Modeling and Verification*.

[38] Use Case, http://en.wikipedia.org/wiki/Use_case, (assessed December 8, 2009).

[39] D. Drusinsky, J.B. Michael and M. Shing, "A Framework for Computer-Aided Validation," *Innovations in Systems and Software Engineering*, 4(2), June 2008, 161–168.

To understand the requirements and constraints for the acquisition process, three use cases were developed to describe the tasks in the concept-refinement phase for the development of the TDS document. Specifically, the use cases describe the process on the interaction between the concept-refinement team, PM, system operators and approval authority. The role of the concept-refinement team is to take the requirement from the ICD and performs analysis to break down the high level requirements into measurable goals. To do this, the concept-refinement team must analyze different alternatives and adopt appropriate strategies to achieve this requirement. Different sets of goals are developed to meet these strategies and for each of these goals, measurable metrics must be defined. The concept-refinement team then develops and submits the draft TDS document to the PM. Upon the receipt of the draft TDS document, the PM will arrange and schedule for a review with the system operators. If the system operator is satisfied with this review, the PM can forward the draft TDS document to the MDA for approval. However, should any of the analysis falls short of the requirement, the PM will task the concept-refinement team to further revise the TDS document and adjust the project schedule accordingly. When the TDS document is finally submitted to the MDA, the MDA is to review and approve the details of the analysis. If the document is approved, the concept-refinement phase is considered complete and the approved TDS document will be used as the input for the next phase of the acquisition process. Otherwise, the PM and his concept-refinement team are required to conduct more analysis based upon the direction given by the MDA. In addition, the PM has to review his project schedule again.

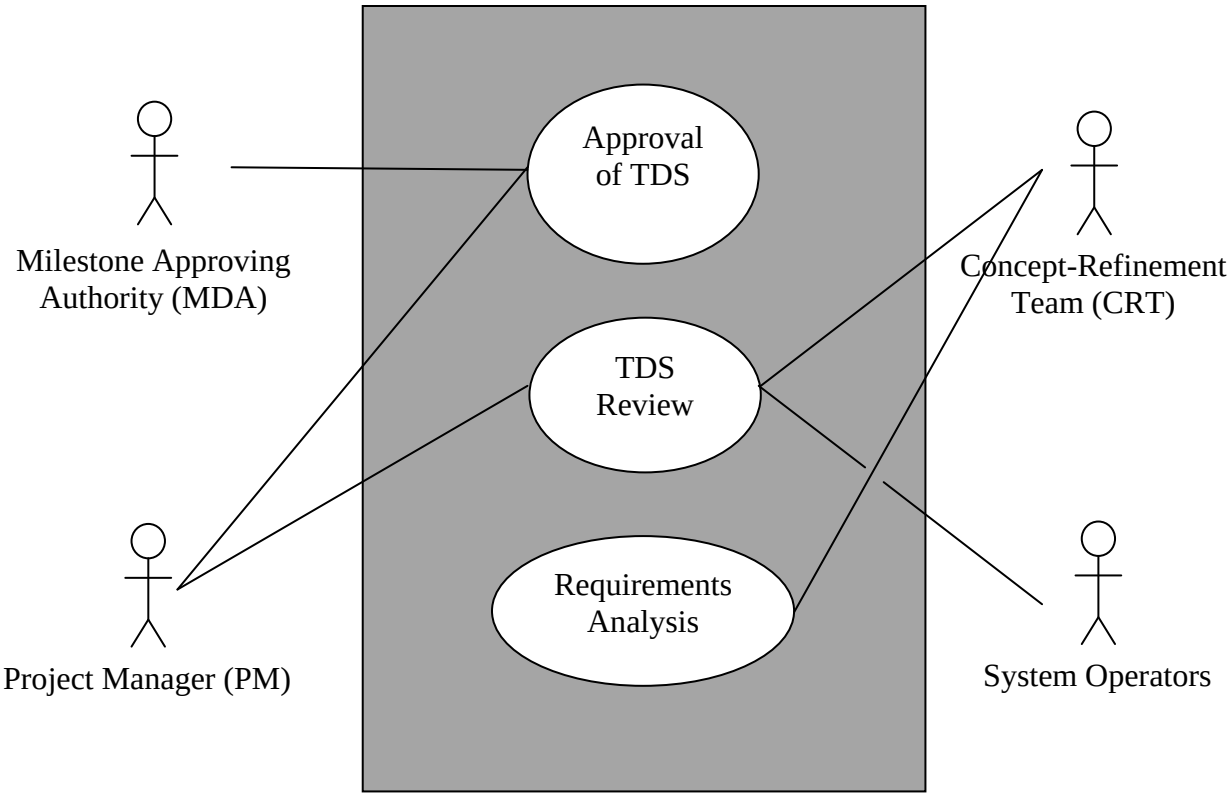

Figure 7. Use Case Model

Figure 7 shows the model of the use cases for the concept-refinement phase. Three use cases that were used as an analysis are as follows:

Use Case: **UC-1 Requirements Analysis**

Primary Actor: Concept-Refinement Team (CRT)

Other Actors: PM

Stakeholders and Interest:

- The CRT needs to consider all alternatives and develop the TDS document.
- The PM wants to complete the concept-refinement phase in time and get the TDS document to be approved by the MDA.

Entry Conditions: Receipt of all the input documents.

Success Guarantee: Produce the TDS document.

Typical Flow of Events:

1. Extract and breakdown requirements from the ICD.
2. For each requirement, consider all the alternatives in the AoA plans.
3. Select strategies to achieve the requirement.
4. Select goals to meet this strategy.
5. Formulate metrics and DRs to measure the performance.
6. Submit the draft TDS document to the PM for review.

Alternate Flows:

1a. Incomplete information to perform analysis.

1. Inform PM of the problem and reset the project schedule.
2. Contact the acquisition sponsor agency and proceed to Step 2 upon receipt of the missing information.

Special Requirements: Thirty days timeline to complete the concept-refinement phase.

Use Case: **UC-2 TDS Review**

Primary Actor: PM

Other Actors: System Operators, CRT

Stakeholders and Interest:

- The PM and the CRT want to review the completed TDS document to ensure it meets the requirements of the system operators.
- The system operators want to ensure that all system specifications are met in accordance to the system requirements.

Entry Conditions: The completed TDS document.

Success Guarantee: The TDS document is reviewed without further analysis required.

Typical Flow of Events:

1. PM contacts and arranges with system operators to review the TDS document.
2. PM forwards the TDS document to the system operators ahead of the review.
3. PM conducts review (attended by the CRT) with the system operators.

4. TDS document is reviewed without further analysis required and is accepted by the system operators.

Alternate Flows:

4a. System operators raised issues with the TDS document.

1. CRT to establish a plan to address the issues and submit revised TDS document to PM for review.

2. PM to revise schedule.

Special Requirements: Thirty days timeline to complete the concept-refinement phase.

Use Case: **UC-3 Approval of TDS**

Primary Actor: PM

Other Actors: MDA

Stakeholders and Interest:

- The MDA wants to ensure that all parameters required in the TDS document are included and the review with the system operators is conducted without any further review required.
- The PM wants to seek approval of the TDS document and complete the concept-refinement phase in time.

Entry Conditions: The reviewed TDS document.

Success Guarantee: The TDS document is approved for the next phase of the acquisition process.

Typical Flow of Events:

1. PM submits TDS document to MDA.

2. MDA conducts review of the TDS document.

3. TDS document is approved.

Alternate Flows:

3a. TDS document is not approved.

1. PM has to create a new task and revise the schedule.

Special Requirements: Thirty days timeline to complete the concept-refinement phase.

2. Workflow Modeling

The next step is to model the workflow in terms of activity diagrams. The flexibility of the activity diagram allows it to be used in a variety of ways. It can be used to describe business flows in varying degrees, complex flows within or between use cases, or complex behaviors within an object.[40] For the author's application, the activity diagram, as shown in Figure 8, is used to describe the flow of activities between different actors during the concept-refinement phase of the acquisition process.

While the actual concept-refinement phase does exist in the U.S. DoD acquisition process, the activities presented in Figure 8 are pieced together based on other research materials; it is just a fictitious representation and not based on an actual project by the U.S. DoD.

As mentioned before, the activity diagram resembles the workflow process of the concept-refinement phase. The key input to kick start this phase is the receipt of the necessary input such as the ICD and AoA plans. The concept-refinement team is required to use these documents to select the best options that meet the requirements to develop the TDS document for the next phase of the program. To do this, the team is to consider various alternatives for each requirement and consider user input/feedback as well. The phase is considered completed when the developed TDS is approved by the MDA.

To elaborate, the concept-refinement phase will commence upon the receipt of the necessary input documents, notably the ICD and the AoA plan. The concept-refinement team is required to consider at least two alternatives as per the AoA plan for each of the requirements that they have selected. Once the alternatives have been considered and the options weighed, the team can proceed to select a strategy for developing this requirement.

[40] D. Drusinsky, J.B. Michael and M. Shing, "A Framework for Computer-Aided Validation," *Innovations in Systems and Software Engineering*, 4(2), June 2008, 161–168.

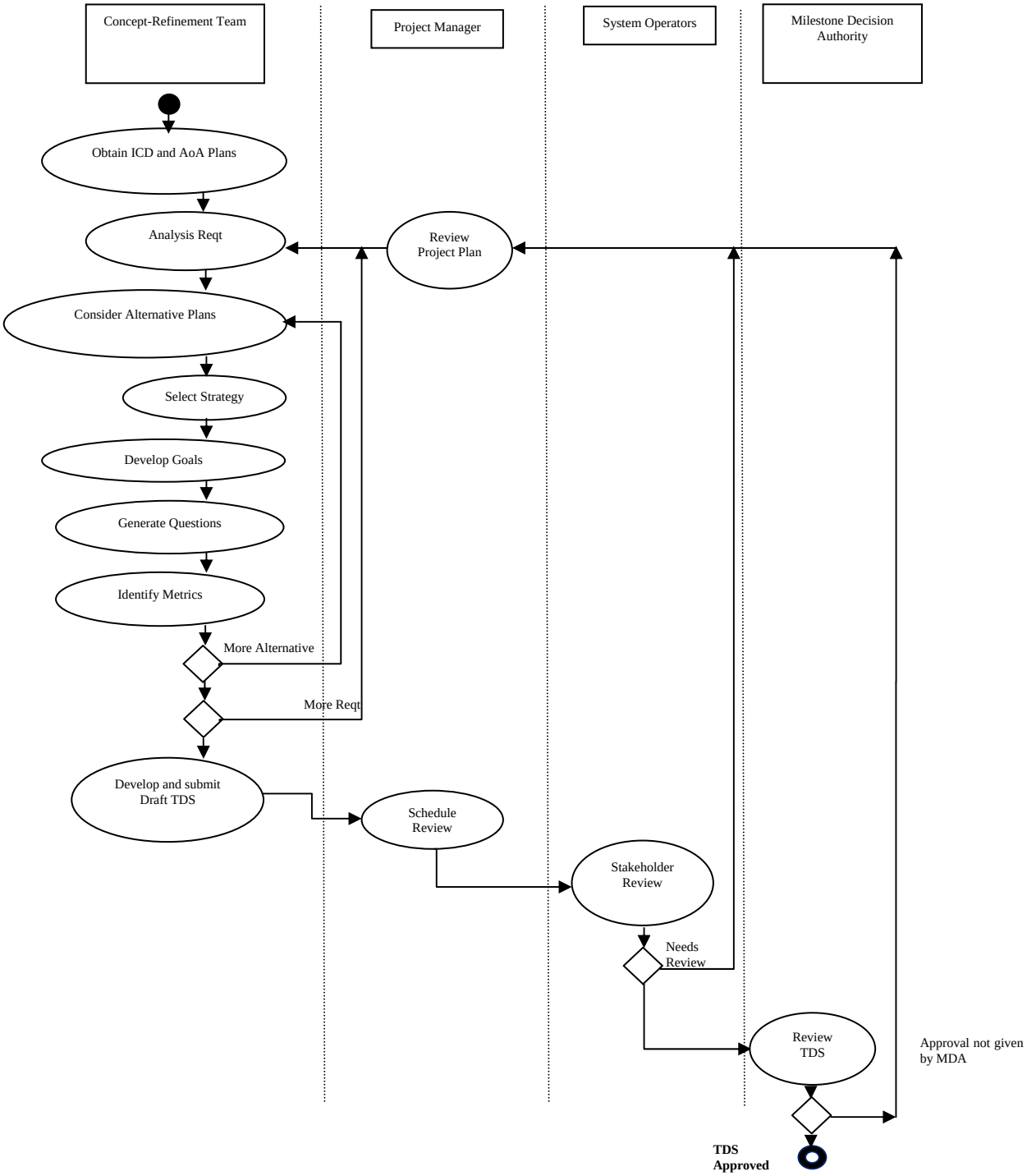

Figure 8. Activity Diagram for Concept-Refinement Phase

Following the GQM model, the goals for the selected strategy must be developed, and questions could be raised as a result of these goals. Finally, the metrics and the DRs can be developed to fulfill the measurement of the respective goals. The process is iterative and considered complete when all requirements are decomposed with the GQM model. Upon completion, the concept-refinement team can proceed to develop the draft TDS document and the PM will arrange with the stakeholders to conduct a review. Depending on the decisions by the stakeholders and project team, it is possible for some requirements to be further analyzed. When this happens, the PM has to revise his project schedule. Otherwise, the TDS document is submitted for approval by the Milestone Decision Authority.

C. STATECHART ASSERTIONS DEVELOPMENT AND VALIDATION

Harel Statecharts[41] are commonly used in the design analysis phase of an object-oriented UML-based design methodology, for example, Brugge suggests using Statecharts in the design analysis phase of an object-oriented, UML-based design methodology to specify dynamic behavior of complex reactive systems.[42] Statechart assertion is a formalism that combines UML-based prototyping, UML-based formal specifications, runtime monitoring, and execution-based model checking.[43, 44] The main advantage for using this methodology is that, unlike temporal logic-based specification languages, which are purely propositional, Statechart assertions are more intuitive to use because they are visual and closer to the thinking of the system designers in the modeling of the system behaviors.

---

[41] D. Harel, "A Visual Formalism for Complex Systems," *Science of Computer Programming* 8, no. 3 (1987): 231–274.

[42] B. Bruegge, *Object-Oriented Software Engineering: Using UML, Patterns, and Java*, 2nd ed. (Prentice Hall, 2004).

[43] D. Drusinsky, "Semantics and Runtime Monitoring of TLCharts: Statechart Automata with Temporal Logic Conditioned Transitions," *Proc. 4th Runtime Verification Workshop (RV 04), Electronic Notes in Theoretical Computer Science* 113, (2005): 3–21.

[44] Drusinsky, *Modeling and Verification.*

The author developed Statechart assertions using the StateRover tool[45], which provides support for design entry, code generation, and visual debugging animation for UML Statecharts combined with flowcharts to provide runtime monitoring and runtime recovery from assertion failures.

Typically, formal specifications are created from a conceptual requirement as understood by the primary modeler. Regardless of the formal notation or method used, system modelers typically begin their requirements discovery process using some scenarios that involve the system and its environment. They first express their understanding of the expected behavior or properties of the system informally using natural language and then translate the assertion into formal assertions following the process shown in Figure 9.[46]

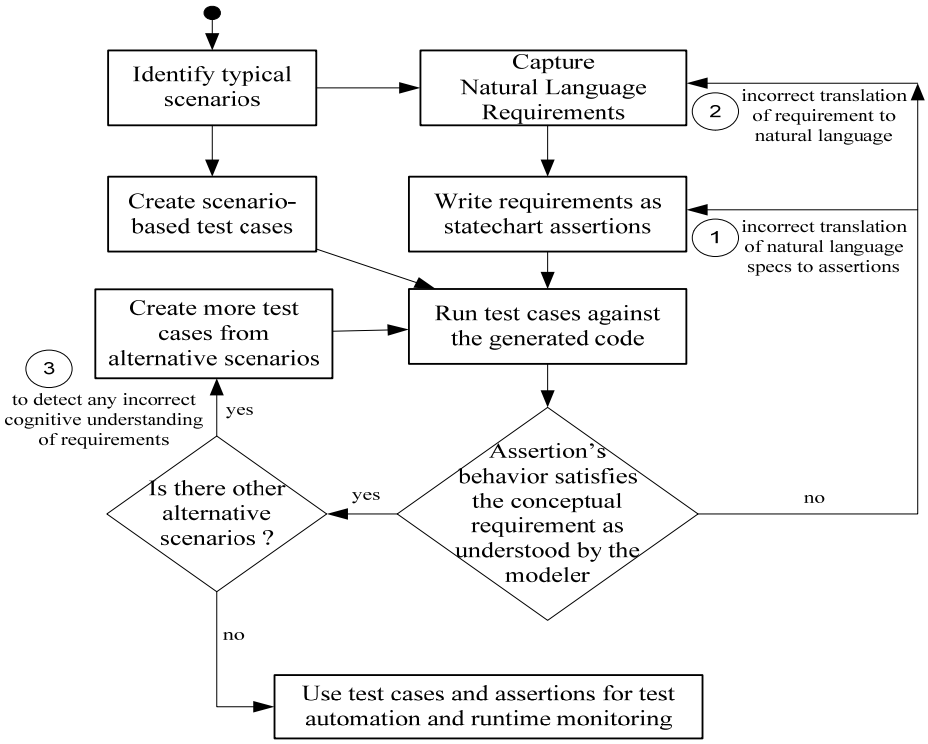

Figure 9. Iterative Process for Assertion Development

[45] Drusinsky, *Modeling and Verification*.

[46] Drusinsky, Shing, and Demir, "Creating and Validating Embedded Assertion Statecharts."

Using the activity diagram in Figure 8, the PM first identifies the events that are relevant to the proper and timely execution of the tasks in the concept-refinement phase as shown in Table 1.

Events	Representation
1. The receipt of the necessary input documents	*Receive input*
2. The analysis of the alternative plans has been considered for each of the requirements stated	*Alternative and requirement selected*
3. Strategies for each requirement are selected	*Strategies selected*
4. The goals are developed	*Goals developed*
5. Questions as a result of the goals are raised	*Questions raised*
6. The metrics are developed	*Metrics developed*
7. The review by the stakeholders are complete	*Review completed*
8. The TDS has been completed	*TDS completed*

Table 1. Events of Interest for Statechart Assertion Model

The PM then proceeds to express, in natural language, the scenarios that describe the proper and timely execution of the tasks based on the observable events, as exemplified by the following two assertions:

Assertion 1 : ***The concept-refinement team must complete the Technology Development Strategy (TDS) document within eight weeks after receiving the necessary documents.***

Assertion 2 : ***The concept-refinement Team must consider at least two alternatives for each requirement.***

Next, the PM translates Assertion 1 into the Statechart assertion shown in Figure 10 and tests the correctness of the assertion using the two test cases shown in Listings 1 and 2.

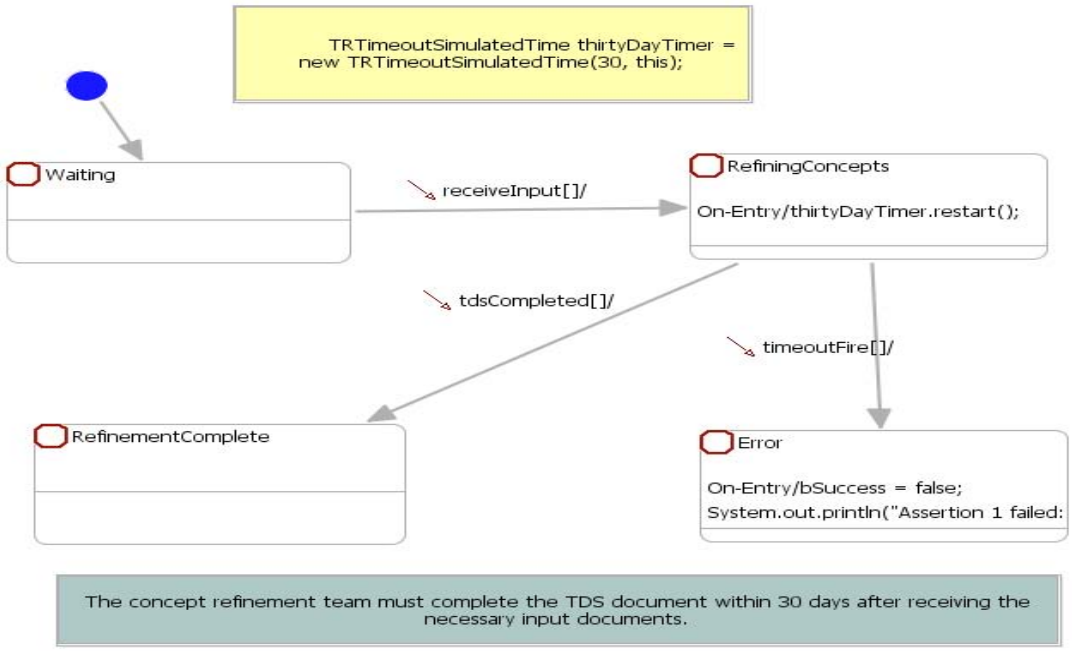

Figure 10. Statechart for Assertion 1

The assertion is written from an observer's point of view who wants to assure that the TDS document can be completed within thirty days from the receipt of the necessary input documents. The Statechart assertion is interested in the events: *receiveInput*, *tdsCompleted* and *timeoutFire*. Upon receipt of the documents, the Statechart assertion will enter the *RefiningConcepts* state. A thirty-day timer will then be fired to keep track of the progress. If the document is completed in a timely fashion, it will enter the *RefinementComplete* state. Otherwise, the assertion would fail and the *timeoutFire* event would lead to an *Error* state. To assure that the assertion works as specified, the JUnit test case in Listings 1 and 2 is used to test the behavior.

```
import junit.framework.TestCase;


public class Assertion1_Test1 extends TestCase {
	private Assertion1 a1 = null;

	protected void setUp() throws Exception {
		super.setUp();
		a1 = new Assertion1();
	}

	protected void tearDown() throws Exception {
		a1 = null;
		super.tearDown();
	}

	public void testAssertion1() {

		a1.receiveInput();
		this.assertTrue(a1.isState("RefiningConcepts"));
		a1.incrTime(29);
		this.assertTrue(a1.isState("RefiningConcepts"));
		a1.tdsCompleted();
		this.assertTrue(a1.isState("RefinementComplete"));
		this.assertTrue(a1.isSuccess());
	}

}
```

Listing 1. Test Case 1 for Assertion 1

Test case 1 is setup to highlight that positive activity has taken place that complied with the assertion statement. It is what the author termed as a “happy” scenario. In this case, the TDS document is completed within thirty days and the Statechart assertion enters the *RefinementComplete* state. Test case 2 represents an exception to the rule, where the time limit of thirty days is exceeded, causing the Statechart assertion to enter the *Error* state, and signaling a violation to the assertion.

```
import junit.framework.TestCase;


public class Assertion1_Test2 extends TestCase {
private Assertion1 a1 = null;

	protected void setUp() throws Exception {
		super.setUp();
		a1 = new Assertion1();
	}

	protected void tearDown() throws Exception {
		a1 = null;
		super.tearDown();
	}

	public void testAssertion1() {
		int i = 0;
		a1.receiveInput();
		this.assertTrue(a1.isState("RefiningConcepts"));
		a1.incrTime(31);
		this.assertTrue(a1.isState("Error"));
		a1.tdsCompleted();
		this.assertTrue(a1.isState("Error"));
		this.assertFalse(a1.isSuccess());
	}
}
```

Listing 2. Test Case 2 for Assertion 1

Figure 11 shows the Statechart assertion for Assertion 2, that the concept-refinement team must consider at least two alternatives for each requirement.

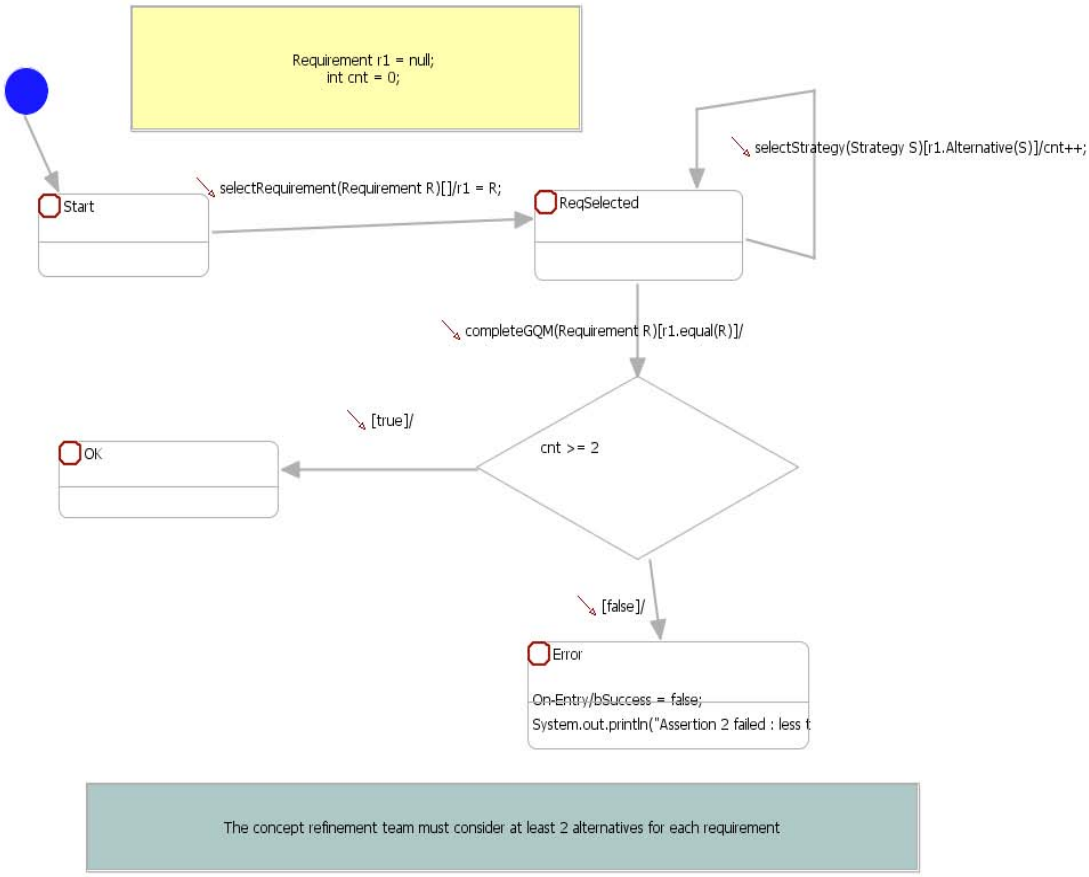

Figure 11. Statechart for Assertion 2

The correctness of the Statechart assertion is validated with three JUnit test cases shown in Listings 3, 4 and 5. Here, the data structure *cnt* is used to keep track of the number of alternatives that were considered with three sequencing events: *selectRequirement*, *selectStrategy*, and *completeGQM*. For every requirement selected, the Statechart assertion will enter the *ReqSelected* state. Within this state, the concept-refinement team will be monitored so that they consider at least two other strategies.

```
import junit.framework.TestCase;

public class Assertion2_Test1 extends TestCase {
……..
	}

	public void testAssertion1() {

		a2.selectRequirement(r);
		this.assertTrue(a2.isState("ReqSelected"));
		a2.selectStrategy(s1);
		this.assertTrue(a2.cnt == 1);
		this.assertTrue(a2.isState("ReqSelected"));
		a2.selectStrategy(s2);
		this.assertTrue(a2.cnt == 2);
		this.assertTrue(a2.isState("ReqSelected"));
		a2.completeGQM(r);
		this.assertTrue(a2.isState("OK"));
		this.assertTrue(a2.isSuccess());
	}
}
```

Listing 3. Test Case 1 for Assertion 2

```
import junit.framework.TestCase;


public class Assertion2_Test2 extends TestCase {
	……….

	public void testAssertion1() {

		a2.selectRequirement(r);
		this.assertTrue(a2.isState("ReqSelected"));
		a2.selectStrategy(s1);
		this.assertTrue(a2.cnt == 1);
		this.assertTrue(a2.isState("ReqSelected"));
		a2.selectStrategy(s2);
		this.assertTrue(a2.cnt == 1);
		this.assertTrue(a2.isState("ReqSelected"));
		a2.completeGQM(r);
		this.assertTrue(a2.isState("Error"));
		this.assertFalse(a2.isSuccess());
	}
}
```

Listing 4. Test Case 2 for Assertion 2

```
import junit.framework.TestCase;


public class Assertion2_Test3 extends TestCase {
      ……….

      public void testAssertion1() {

            a2.selectRequirement(r);
            this.assertTrue(a2.isState("ReqSelected"));
            a2.selectStrategy(s1);
            this.assertTrue(a2.cnt == 1);
            this.assertTrue(a2.isState("ReqSelected"));
            a2.completeGQM(r);
            this.assertTrue(a2.isState("Error"));
            this.assertFalse(a2.isSuccess());
      }
}
```

Listing 5. Test Case 3 for Assertion 2

These three scenarios were similar to the test scenarios of the first Statechart assertion. Here, test case 1 represented a typical "happy" scenario where all requirements are adhered to; that is, for every requirement, at least two alternatives were considered. For the second test case, although there were two strategies, they belonged to two separate requirements. Upon observing the complete *completeGQM* event, the Statechart assertion exits the *ReqSelected* state. The data structure *cnt* that keeps track of the number of strategies selected will note that in this case only one strategy was selected for each requirement, and they will transit to the *Error* state. For the third case, it is straightforward as only one strategy was selected before attempting to exit the *ReqSelected* state. Similarly, the assertion failed as expected and the *Error* state was entered.

D. APPLICATION OF THE STATECHART ASSERTION FOR THE RUNTIME MONITORING OF THE ACTUAL ACQUISITION PROCESS

The Statechart assertions, which were developed for the concept-refinement phase, can be applied to the monitoring of the actions taken by the concept-refinement team. To achieve this, a protocol would be established between the PM and the concept-refinement team. The protocol would describe the events of interest, such as those described in Table

1, as well as the procedure to follow for the concept-refinement team to notify the PM of the occurrence of these events, which are time-stamped and recorded in an event log. Whenever the log file is updated, the time-stamped event sequences in the log either are translated manually, or using an automated tool, into JUnit test scenarios like those shown in Listings 1-5 and run against the executable assertions. The PM will be notified whenever an event sequence violates any of the Statechart assertions.

THIS PAGE INTENTIONALLY LEFT BLANK

V. CONCLUSION

A. SUMMARY

In this thesis, the author addressed the challenge to formally specify and monitor the workflow for the process to acquire a complex system. The first step in the approach is to apply the GQM method to identify the program goals and develop metrics and DRs to measure the degree of success in accomplishing the goals. The next step is to build activity diagrams to model the workflow process. The artifacts generated in these steps are used to guide the formal specification of the workflow. In this thesis, the author used Statechart assertions as the formalism and the StateRover tool to demonstrate the technical feasibility of validating the workflow specification and the tools used in conducting automated routine execution monitoring.

The acquisition process for complex systems will continue to evolve and reform. What remains invariant is the need for assurance that the process is faithfully executed. In this regard, the author modeled a part of an acquisition process to illustrate that program requirements can be specified and validated using Statechart assertions. It is acknowledged that the Statechart assertions and the test cases presented here are very simple and could be conducted through manual examination of requirements and design artifacts. However, this simple example did demonstrate the use of Statechart assertion as a means of enforcing process requirements. The test cases also demonstrates the checking of decision-making and the adherence of requirements in the context of the modeled process.

While the Statechart assertions were not applied to other phases of the acquisition process, the technique could still be valid in assuring the process requirements in other phases. In addition, this technique can also aid developers in demonstrating that their software satisfies the requirements (functional and nonfunctional), and effectively locates and explains the cause of errors in faulty design and development. In most complex systems, for example, building an air defense system correctly that meets the operational needs of the battalion, a manual V&V technique is almost inadequate. Most of it is due to

the fine level of detail during runtime that makes human intervention almost impractical. Fortunately, the utilization of the Statechart assertions and the StateRover tool provided valuable runtime validation of behavioral requirements. This allows the stakeholders to capture the formal requirements to ensure that the developer's cognitive understanding of the requirements matches the formal specifications.

B. FUTURE WORK

More work is needed to explore the application of the approach to the rest of the acquisition process.

1. The Statechart assertions and the JUnit test case scenarios developed using the StateRover can be extended beyond the concept-refinement phase to the other phases of the acquisition process. The test scenarios can be integrated to form a complete system-acquisition model. Once completed, it is necessary to include a suite of validation test scenarios to ensure the correctness of the formal Statechart assertions.
2. While GQM can facilitate requirements analysis and guide the development of metrics to measure the degree to which program goals are met, it is important to develop the complete framework to use the metrics to drive the follow-on efforts and measure the success of the acquisition program.
3. The process of creating Statechart assertions from the activity diagram can be rather haphazard because it relies on the expertise of the modeler. One way to make the approach more structured and systematic is to develop templates and patterns for identifying natural language requirements from the UML activity diagrams.[47] Another way to reduce the burden on the modelers and to improve the reliability and assurance of the assertions is to develop libraries of pre-tested generic Statechart assertions accompanied by scenario-based test cases.[48] Templates and reusable assertions for the acquisition process need to be developed. It may be possible to improve the quality and efficiency of an acquisition by equipping developers with a library of templates and generic assertions.
4. The author proposed to use runtime execution monitoring to assure the proper and timely execution of the acquisition process. The artifact involved was the log file containing the time-stamped events reported by

[47] D. Drusinsky, "From UML Activity Diagrams to Specification Requirements," *Proc. IEEE International Conference on System of Systems Engineering (SoSE '08)*, June 2-4, 2008, 1–5.

[48] D. Drusinsky, J. B. Michael, T. W. Otani and M. Shing, "Validating UML Statechart-Based Assertions Libraries for Improved Reliability and Assurance," *Proc. 2nd International Conference on Secure System Integration and Reliability Improvement (SSIRI '08)*, July 14-17, 2008, 47–51.

the project members. It was noted that to execute the test scenarios, the log file had to be translated to the JUnit test cases. Manual translation is not only inefficient (as the author can expect a lot of such translations in an actual acquisition environment), but it also relies heavily on expertise intervention in preparing the test cases. The resultant JUnit test cases also need to be tested to ensure that they are working correctly. As such, a one-time developmental effort should be invested in to design and implement a tool to automate the collection of the time-stamped events, the translation of the event log into a JUnit test case, and the execution of JUnit test against the Statechart assertions. A report could subsequently be generated to inform the PM on the progress of the processes so as to provide runtime monitoring and recovery from assertion failures. The only thing requiring manual input would be given by the team members upon the completion of their tasks. One goal the developer should seek is to automate the processes of organizing, retrieving information from, and interfacing the libraries as much as possible in an attempt to reduce errors.

THIS PAGE INTENTIONALLY LEFT BLANK

LIST OF REFERENCES

Basili, V. R. 1981. Data Collection, Validation, and Analysis. *Tutorial on Models and Metrics for Software Management and Engineering*, IEEE Catalog no. EHO-167-7. 310–313.

Basili, V. R., and Rombach, H. D. 1988. The TAME Project: Towards Improvement-Oriented Software Environments. *IEEE Transactions on Software Engineering* SE-14, no. 6: 761.

Bennett, Simon, Skelton, John, and Lunn, Ken. *Schaum's Outlines of UML*. McGraw-Hill International, 2001.

Bruegge, B. *Object-Oriented Software Engineering: Using UML, Patterns, and Java*, 2nd ed. Prentice Hall, 2004.

Carnegie-Mellon University. "Acquisition Overview: The Challenges." https://buildsecurityin.us-cert.gov/daisy/bsi/articles/best-practices/acquisition/893-BSI.html (accessed December 8, 2009).

Chen, H., and Wagner, D. 2002. MOPS: An infrastructure for examining security properties of software. *CCS '02: Proceedings of the 9th ACM Conference on Computer and Communications Security*, pp. 235–244.

Creese, S. 2001. Data Independent Induction: CSP Model Checking of Arbitrary Sized Networks. D.Phil. thesis, Oxford University, 2001.

Defense Acquisition University (DAU). "Integrated Defense Acquisition, Technology, and Logistics Life Cycle Management System." June 15, 2009. https://acc.dau.mil/IFC/index.htm (accessed November 11, 2009).

Department of Defense (DoD). Directive 5000.1. "The Defense Acquisition System." May 12, 2003.

Department of Defense (DoD). Instruction 5000.2. "Operation of the Defense Acquisition System." May 12, 2003.

Diimitrios, G., Mark, H., and Amit, S. 1995. An overview of workflow management: From process modeling to workflow automation infrastructure. *Distributed and Parallel Databases* 3, no. 2, pp. 119–153.

Drusinsky, D. 2005. Semantics and Runtime Monitoring of TLCharts: Statechart Automata with Temporal Logic Conditioned Transitions. *Proc. 4th Runtime Verification Workshop (RV 04), Electronic Notes in Theoretical Computer Science*, vol 113, pp. 3–21.

Drusinsky, D. 2008. From UML Activity Diagrams to Specification Requirements. *Proc. IEEE International Conference on System of Systems Engineering (SoSE '08)*, pp. 1-5.

Drusinsky, D. *Modeling and Verification Using UML Statecharts A Working Guide to Reactive System Design, Runtime Monitoring and Execution-based Model Checking*. Burlington, MA: Elsevier, 2006.

Drusinsky, D., Michael, J. B., Otani, T. W., and Shing, M. 2008. Validating UML Statechart-Based Assertions Libraries for Improved Reliability and Assurance." *Proc. 2nd International Conference on Secure System Integration and Reliability Improvement (SSIRI '08)*, pp. 47–51.

Drusinsky, D., Shing, M., and Demir, K. 2007. Creating and Validating Embedded Assertion Statecharts. *IEEE Distributed Systems Online*, 8, no. 5, art. no. 0705–o5003.

Emmerich, W., and Gruhn, V. 1991. "FUNSOFT nets: a Petri-net based software process modeling language." *Software Specification and Design, Proceedings of the Sixth International Workshop*, pp. 175–184.

GlobalSecurity.org. http://www.globalsecurity.org/military/systems/munitions/claws.htm (accessed November 15, 2009).

Harel, D. 1987. A Visual Formalism for Complex Systems. *Science of Computer Programming* vol 8, no. 3, pp. 231–274.

Jackson, D. *Software Abstractions: Logic, Language, and Analysis*. MIT Press, 2006.

Net Resources International. "Surface-Launched AMRAAM (SL-AMRAAM / CLAWS) Medium-Range Air Defense System, USA." http://www.army-technology.com/projects/surface-launched/surface-launched1.html (accessed November 15, 2009).

Roscoe, A. W. *The Theory and Practice of Concurrency*. Great Britain: Prentice Hall Europe, 1997.

Schumann, M. A. 2009. A Statechart Model of the Cross Domain Implementation Process. *IATAC Newsletter* 12 (February 2009), pp. 26–30.

Software Engineering Institute. Carnegie-Mellon University. "Capability Maturity Model Integration (CMMI)." http://www.sei.cmu.edu/cmmi/start/faq/related-faq.cfm (accessed December 8, 2009).

Solingen, Rini, and Berghout, Egon. *The Goal/Question/Metric Method: A Practical Guide for Quality Improvement of Software Development*. McGraw-Hill, 1999.

Van Solingen R., Van Latum, F., Oivo, M., and Berghout, E, 1995. Application of Software Measurement at Schlumberger RPS: Towards Enhancing GQM. *Proceedings of the 6th European Software Control and Metrics (ESCOM) Conference*, pp. 17-19..

Wikipedia. "Alloy Analyzer." http://en.wikipedia.org/wiki/Alloy_Analyzer, (accessed December 8, 2009).

Wikipedia. "Process Modeling." http://en.wikipedia.org/wiki/Process_modeling, (accessed December 8, 2009).

Wikipedia. "Use Case." http://en.wikipedia.org/wiki/Use_case, (accessed December 8, 2009).

THIS PAGE INTENTIONALLY LEFT BLANK

INITIAL DISTRIBUTION LIST

1. Defense Technical Information Center
   Ft. Belvoir, Virginia

2. Dudley Knox Library
   Naval Postgraduate School
   Monterey, California

3. Professor Peter Denning
   Naval Postgraduate School
   Monterey, California

4. Professor James Bret Michael
   Naval Postgraduate School
   Monterey, California

5. Professor Man-Tak Shing
   Naval Postgraduate School
   Monterey, California

6. Professor Yeo Tat Soon, Director
   Temasek Defence Systems Institute
   National University of Singapore
   Republic of Singapore

7. Ms Tan Lai Poh, Assistant Manager
   Temasek Defence Systems Institute
   National University of Singapore
   Republic of Singapore